ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2012

Two Level Nested Loops Tiled Iteration Space Scheduling By Changing Wave-Front Angles Approach

AliReza Hajieskandar, Shahriar Lotfi, Simin Ghahramanian

Department of Electrical and computer engineering, Islamic Azad University, Iran
Department of Computer Science, University of Tabriz, Tabriz, Iran
Sama Technical and Vocational Training College, Islamic Azad University, Iran

ABSTRACT— ***Most of scientific applications need high speed and powerful computations. Using the computational potential of multiple processors enables us to achieve this goal. This factor as well as the story of former sequential programs that were so costly in generation, culminated at the invention of a tool named as "super-compiler" to automatically convert sequential code into parallel code. Super compilers identify the hidden parallelism inside programs and convert sequential programs into parallel ones. As most of computational programs incorporate nested loops, the parallel execution of these loops increases the running speed of programs. So the parallelization of nested loops is crucial for increasing the execution speed of programs. One of the conversion stages of sequential nested loops into parallel ones is to schedule the tiled iteration space. Regarding the fact that, so far the block and cyclic approaches have been introduced , in this paper the wave-front approach and wave-angle shifts have been incorporated in the block and cyclic approaches in order to reduce the execution time of two-level nested loops.***

Keywords— **nested loops, Iteration Tiled Space, Scheduling, Wave Fronts.**

I. INTRODUCTION

Many computational programs use nested loops. Programs run faster if the loops run in parallel. Once upon a time in past, writing these programs were very costly. In order to avoid these costs a tool named as "super-compiler" is used to convert these programs to parallel ones. . Therefore nested loops parallelization is a major focus for fast programs. The conversion of nested loops into parallel ones is accomplished through the following orderly stages: In stage I, as two instructions can run in parallel only when there is no data dependency between them, the data dependencies in nested loops will be analyzed and extracted accordingly. Data dependencies originate from the arrays used inside loops .Therefore if the instruction S executes before the instruction T and generates some data that the instruction T will use it certainly, T is called a data dependent or shortly a dependent on S. In stage II, in order to achieve better parallelization, to decrease inter-processor connections and hence to optimally distribute the dependent iterations of nested loops for being executed in processors, iteration space is tiled. A set of loop iterations running in one processor is called a tile. In stage III, based on the shape and size of produced tiles a desired parallel code is generated for the iteration space of stage II. Finally in the stage IV the tiled space of the former stage is scheduled by using the wave-front approach. In fact, the tiles are assigned to the processors in a way that the time needed for executing all of them is minimized accordingly. In other words termination time for the last tile of final wave is shortened as much as possible. This paper focuses mainly on the fourth stage, parallelization.

The remainder of this paper is organized as follow: in Section II Scheduling problem definitions was presented. In section III we review the basic concepts of the problem in question. Section IV explains former related works in the context. Section V describes our proposed approach. Section VI and VII focus on the assessment of experiments and conclusions as well as a guideline for future works respectively.

II. THE PROBLEM

In this section, inspired by an example, I will explain the issue of scheduling the tiled iteration space for nested loops. Consider the nested loops of following example:

```
For i:= 0 To 7 Do
  For j:= 0 To 5 Do
      A[i, j]:= A[i-1, j] + A[i, j-1];
  EndFor
EndFor
```

Reference to array in this example, creates data dependency among loop iterations. After analyzing the data dependency, the iteration space can be tiled in order to reduce inter-processor connections. In fact a tile is a set of iteration points with many data dependencies implying the need for being executed in one processor[12]. So the iteration space $J^S = \left(J_1^S, J_2^S\right)$ and the tiled space of the mentioned example is like Fig.1 in which points represent loop iterations while directed edges account for data dependencies as well as their dependent source and destination.

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2012

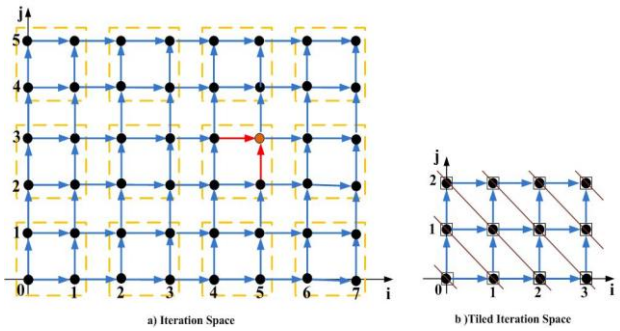

Fig. 1 Iteration space for above nested loop and it's Tiled Iteration Space

In this problem we want to assign a certain number of processors to the existing tiles in the tiled space, so the inputs of our problem are the tiled iteration space and a certain number of specified processors. In this tiled space, each tile $J^S = (J_1^S, J_2^S)$ has two prerequisite tiles as $J'^S = (J_1^S - 1, J_2^S)$ and $J''^S = (J_1^S, J_2^S - 1)$. It means that in the related tiled space there is an edge from two prerequisite tiles toward the corresponding tile. So while the execution of prerequisite tiles has not been finished completely, J^S cannot start execution.

For optimum scheduling of tiled space two constraints are postulated: (1) balanced loads among processors (2) The least possible cost for inter-processor connections when scheduling. The nearly identical figures of tiles (tiles are created incompletely in borders) and the execution of each tile in one processor satisfies the first constraint. In order to reduce inter-processor connecting cost, the tiles with higher communication cost should be assigned to the same processors so that with the zero connection time, the time needed for executing tiles minimizes.

We denote the cost of communication and message sending between the current tile and the prerequisite ones with Vcomm . Since each tile connects to two tiles of different dimensions, so we use two parameters as Vcomm(1) and Vcomm(2) . Vcomm(1) stands for the communication volume of J^S with the neighboring tile in respect for the first dimension (J'^S) while Vcomm(2) stands for the communication volume of J^S with the neighboring tile for the second dimension (J''^S). As long as J^S and its neighboring tiles are executed over different processors we have to tolerate the existing costs as the problem input but for the case of identical processors , the communication cost will be zero. Regarding the fact that processors are fully-connected in style; i.e. each processor is directly connected to other processors, the connection cost among all processors is identical.

The other input of problem i.e. Vcomp accounts for the execution cost of each tile in an individual processor. Given that this parameter is identical for the execution of all processors the termination time of tile J^S is computed through the following equation.

$$completiontime(J^S) = Max\begin{pmatrix} completiontime(J'^S) + V_{comm}(1), \\ completiontime(J''^S) + V_{comm}(2) \end{pmatrix} \quad (1)$$

The total time duration for executing all current scheduled tiles, known as "makespan" is given by the following equation:

$$makespan = \max(completion\ time(J^S)) \quad , J^s \in Tiled\ IterationSpace \quad (2)$$

The underlying objective of tiled space schedule is to find a scheme for assigning the available processors to the tiles which have the shortest makespan time. So the output of problem is an assignment scheme for the tiled iteration space and its makespan.

III. DATA DEPENDENCY AND THE NESTED LOOPS ITERATION SPACE TILING, AN OVERVIEW

The first stage in parallelization of nested loops is to analyze the data dependency problems; two instructions can be executed simultaneously only if there is no data dependency between them. Generally speaking the T instruction is called S - data dependent or shortly S-dependent if both of instructions refer to the same memory location, S instruction is executed before T instruction, if there is an execution path between the two instructions and in the time interval between S and T instructions nothing is recorded in the commonly referred location.

In order to achieve better parallelization and cache locality of reference in individual processors and extraction of great parallelization in multiprocessors the iteration space of nested loops is tiled. A set of loop iterations which must run in an individual processor is called a tile. The purpose from tiling is to reduce communication volume among processors and consequently to optimally distribute the dependent nested loops for being executed in the processors which exchange messages with each other. A sample tiling method is depicted in the below diagram [4].

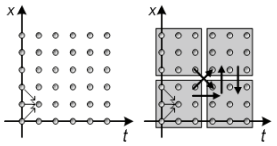

Fig. 2 An example of Iteration Space Tiling

IV. RELATED WORKS

Many approaches have been introduced for scheduling of the iteration space of nested loops each somehow trying to reduce the execution time of loop iterations in a parallel manner [1, 6, 8, 11, 13, 15, 16, 18, 19, 21, 22, 24 and 25]. In

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2012

some methods, in order to minimize the overall execution time of iterations, the communication time of processors overlaps with the time of internal calculations of processors [3, 7 and 17]. Also some other approaches inspired by the wave-front method, eliminate the need for communicating of many processors and assign dependant caches to the same processor. In other approaches, the critical path for the task graph resulted from the loop iterations' space is calculated and in order to obtain load balancing, other non-critical iterations are distributed among processors according to their data dependency. Actually, the critical path includes all loop iterations which should be executed in a certain time. In this procedure, the earliest time of loops' execution as well as the latest makespan of loop iterations are computed and the order of iterations over the critical path is specified accordingly. In this procedure the layers of critical path is specified in which a sum of executable iterations exist in parallel. The non-critical iterations are distributed among processors to preserve load balancing.

Andronikos et. al. in [1] suggested a multinomial algorithm for efficient scheduling of uniform dependency loops by outlining precise criteria for a number of engaged processors in which they identified upper and lower limits for optimum number of required processors which minimizes the total execution time and then performs a binary search between the two borders.

Most of approaches which concentrated on optimum scheduling originate from hyper-plane methods. The main objective of these methods is to classify computations into well-defined groups called wave or hyper-plane by using linear transformations .All iterations existing inside a loop can be executed simultaneously. Darte in [5] showed the optimality of this method. Moldovan, Shang and Darte et al benefited the hyper-plane method to explore an optimal linear schedule by using Diophantine equations, linear programming in subspaces [23] and accurate programming [5]. Hence more computations will be rendered in parallel.

Finally, some approaches try to reduce the overall duration time of parallel execution for nested loops by changing wave angles. In these approaches, the appropriate angle for waves is calculated in a way that the required number of processors for tiles' parallel execution never exceeds the total sum of processors.

Athanasaki et. al. in [2] introduced the following four methods of the cyclic assignment schedule, the mirror assignment schedule, the cluster assignment schedule and the retiling schedule for scheduling of tiled iteration space to clustered system with a given number of SMP(Symmetric Multi-Processors) nodes and applied two execution schemes including the overlapping execution scheme and non-overlapping execution scheme.

Doritos et. al. in [7] introduced a low-cost algorithm which using geometrical calculations generates the typical schemes consisting of iteration subsets which can be executed earlier in a certain k-time phase.

Lotfi and Parsa in [20] suggested a new executable algorithm over non-rectangular n-dimensional tiles in irregular iteration spaces for deriving and scheduling of parallel waves. In order to derive efficient parallel waves they assumed that all tiles with identical coordinate sums locate in one wave and they used the modified block scheduling for assigning parallelepiped tiles of every wave to different processors.

We in the former articles [9 and 10] using the wave-front method introduced two genetic-based algorithms. One of the methods with angle shift and the other without it can culminate at the proper scheduling of the problem in question.

V. PROPOSED STRATEGY

The proposed approach of BCAALS[1] is based on the wave-front method and works by incorporating the wave notion into the BlockH[2], the BlockV[3], the CyclicH[4], the CyclicV[5] methods and due alterations trying to minimize the makespans of nested loops' execution. In fact, a wave consists of tiles without data dependency which can be executed in parallel. If waves are drawn over the iteration loop, they form an angle which its shift culminates at the minimization of required processors in number, the optimization of their loading balance and consequently the optimization of makespan for scheduling. In the rest, the calculation method for makespans by adding the angle concept into the wave-front method and the proposed algorithm is discussed with an example.

A. The pseudo code for computing the overall execution time of tiles in a tiled space based on the wave-front method and inspired by the angle concept

To do this the pseudo code of Figure 3 is used. This algorithm is based on the wave-front method. The waves should be executed sequentially. So in this algorithm the overall execution time equates he sum of tiles' execution times over waves. Presumably, at the wave beginnings processors are synchronized and between two successive waves the transmission time of messages are overlapped possibly. Also for identifying the number of wave in which the tile $J^S = \left(J_1^S, J_2^S\right)$ is located, the Relation No.3 is used in which a and b stand for angle coefficients. (The wave angle is represented by two coefficients of a and b) These coefficients were taken as 1 in the former works which account for a 45 degree angle indeed.

[1] Block and Cyclic Algorithm with Angle for Loop Scheduling
[2] Horizontal Block
[3] Vertical Block
[4] Horizontal Cyclic
[5] Vertical Cyclic

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2012

$$\text{Number of Wave-front}(i,j) = a \times i + b \times j \qquad (3)$$

Scheduling Completion Time Algorithm:

```
Scheduling Completion time := 0
ForEach wave front number, wn, lwn ≤ wn ≤ hwn Do
   ForEach Processor Pi, 0 ≤ i ≤ N −1 Do
      Processing time (Pi )= 0;
   End ForEach
   ForEach tile J^s=(j^s_1, j^s_2) on wave front wn such that a. j^s_1 + b. j^s_2 = wn Do
      find processor number, Pi, assigned to j^s
      assume J's =(j^s_1-1, j^s_2) and J''s=(j^s_1, j^s_2-1)
      find processor number, pi' and pi'' assigned to J's and J''s, respectively
      /* computation time of the tile */
      Processing time (pi) := Processing time (pi) + Vcomp . Tcomp
  (1) If J^s is only dependent on J's Then
        If pi ≠ pi' Then /* communication time(1)*/
          Processing time (pi) := Processing time (pi) + Vcomm(1) . Tcomp
        EndIf
  (2) Else If J^s is only dependent on J''s Then
        If pi ≠ pi'' Then /* communication time(1)*/
          Processing time := Processing time (pi) + Vcomm(2) . Tcomp
        EndIf
  (3) Else If J^s is dependent on both J's and J''s Then
        If pi' ≠ pi'' Then
          If pi ≠ pi', pi'' Then
            Processing time:= Processing time (pi) + max(Vcomm(1) . Tcomm , Vcomm(2) . Tcomm)
          EndIf
        Else If pi ≠ pi', pi'' Then
          Processing time(pi) := Processing time (pi) + Vcomm(1) . Tcomm + Vcomm(2) . Tcomm
        EndIf
      EndIf
    EndIf
  EndIf
  Processing time (wn) = max (processing time (pi)),   ∀i=0, 1,..., N-1
  Scheduling completion time := Scheduling completion time + Processing time (wn)
  EndForEach
EndForEach
```

Fig. 3 The pseudo code for computing the overall execution time of tiles in a tiled space

As observable, there are four kinds of tiles: 1- tiles located in the origin of coordinates 2- tiles located alongside the horizontal axis 3- tiles located alongside the vertical axis 4- internal tiles.

B. An explanatory example for the proposed algorithm

In order to get a better understanding consider the following example and its iteration space.

Example 2:

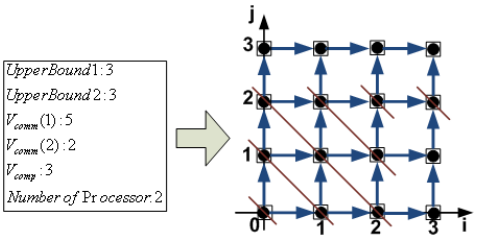

Fig. 4 An Example with its corresponding Tiled Iteration Space

To analyze the problem and discuss the proposed algorithm first we test the above example with the horizontal

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2012

Block , Vertical Block , the Horizontal Cyclic and Vertical Cyclic(CyclicV) methods (Fig 5-a). Then we incorporate the angle concept which in fact is our proposed approach. In part b of Figure 5, the angle waves are shifted which in fact is our proposed algorithm.

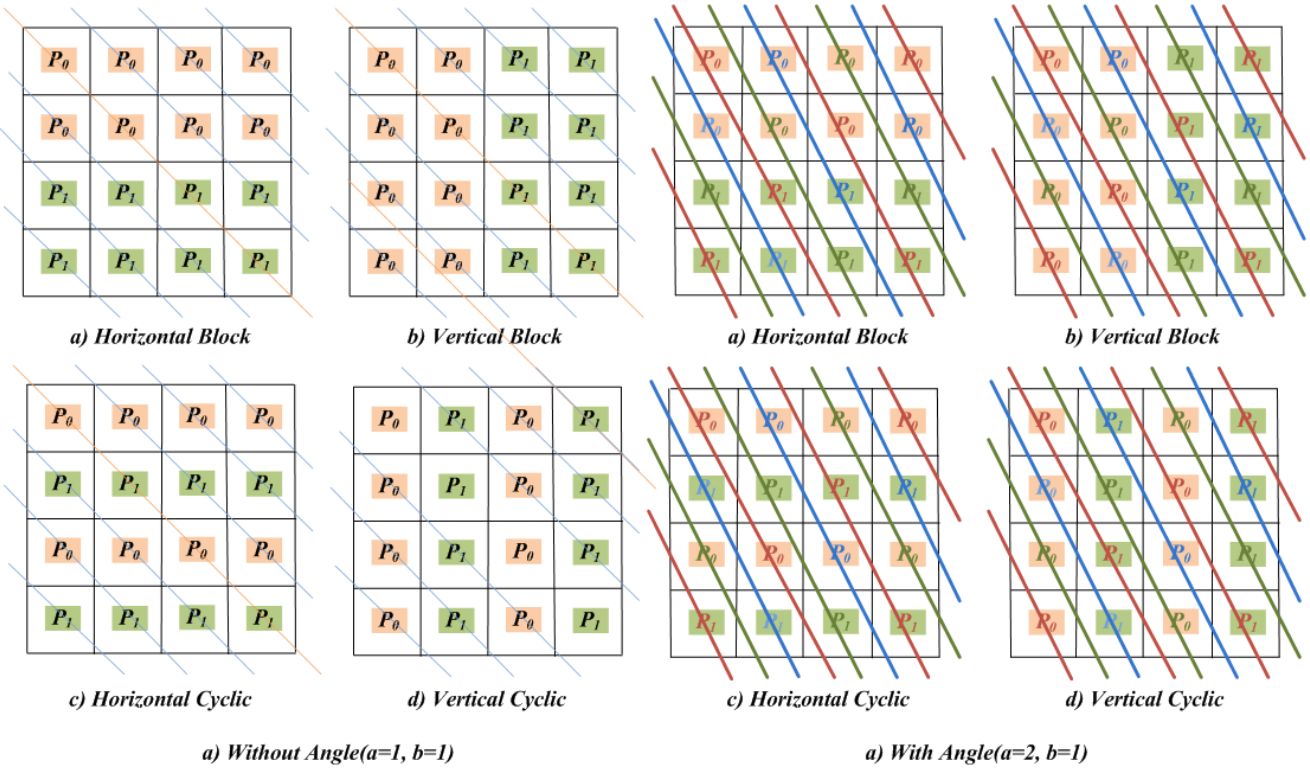

Fig. 5 Previous and proposed Strategy

As noted before, two constrains should be postulated for optimum scheduling of tiled space (1) loading balance among processors and (2) low communication cost among processors in scheduling time. The total sum of the present tiles over the largest wave in Figure 5 reveals that this number is 4 for the former approach while it is 2 for the current proposed approach. Therefore one can conclude that wave angle shift decreases the required number of processors for scheduling the tiled iteration space which in turn decreases the loading balance among processors. So changing wave angles satisfies the first constraint and similarly the second constraint is automatically captured by the engagement of the four mentioned methods because they are designed in nature to reduce communication volume among processors when scheduling.

VI. ASSESSMENT AND PRACTICAL RESULTS

In order to verify the proposed algorithm BCAALS and to contrast its results versus the former algorithms, a software application is designed and implemented in Delphi 7. The engaged algorithms for comparing the new proposed algorithm include the horizontal Block, the Vertical Block, the Horizontal Cyclic and the Vertical Cyclic (CyclicV) algorithms.

A. Proposed Algorithm assessment

As the proposed algorithm is definitive (not random), and its yielding solutions are the same for a given problem, so the proposed algorithm is completely (100%) stable and the generated solutions are also 100% reliable.

B. The comparison of the proposed algorithm with former works

In this section, by conducting various tests of mixed types, the quality of generated solutions by the proposed algorithm is compared against those of former ones. Notably, because of page paucity, only some of tests are shown here. The testing results as well as the related parameters of each test are tabulated in Table I.

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2012

TABLE I

THE OUTCOME RESULTS FROM IMPLEMENTING THE PROPOSED ALGORITHM AND FORMER ALGORITHMS

Experiment Number	Input Parameters						Algorithms														
							Previous Methods				Proposed strategies										
	NP	UB1	UB2	Vcomp	Vcom1	Vcom2	BlockV	BlockH	CyclicV	CyclicH	BlockV	a	b	BlockH	a	b	CyclicV	a	b	CyclicH	a	b
1	3	14	5	10	1	1	728	396	372	396	407	1	5	356	2	1	372	1	1	356	2	1
2	4	11	5	10	1	1	460	333	284	333	278	1	3	293	2	1	284	1	1	293	2	1
3	2	14	5	10	1	1	814	553	525	553	556	1	7	495	3	1	525	1	1	495	3	1
4	2	7	3	10	2	5	286	235	212	235	208	1	4	220	2	1	212	1	1	220	2	1
5	3	7	3	10	2	5	252	235	188	235	180	1	3	220	2	1	188	1	1	220	2	1
6	5	15	9	10	2	5	910	585	476	585	544	1	4	495	2	1	476	1	1	495	2	1
7	4	9	3	8	5	5	272	164	164	164	182	1	3	164	1	1	164	1	1	164	1	1
8	3	9	3	8	5	5	286	237	250	237	201	1	4	226	2	1	250	1	1	226	2	1
9	2	9	3	8	5	5	303	237	276	237	220	1	5	226	2	1	276	1	1	226	2	1
10	2	5	3	8	6	4	178	148	184	148	144	1	3	136	2	1	184	1	1	136	2	1
11	4	9	14	8	6	4	620	712	654	712	518	1	3	456	4	1	654	1	1	456	4	1
12	3	9	14	8	6	4	730	844	786	844	624	1	4	524	5	1	786	1	1	524	5	1
13	3	9	5	2	10	10	174	174	268	174	120	3	4	120	4	3	268	1	1	120	4	3
14	5	9	14	2	10	10	300	316	436	316	256	1	2	214	3	1	436	1	1	214	3	1
15	2	14	7	2	10	10	266	290	728	290	222	1	7	240	3	2	684	5	6	240	3	2
16	3	7	3	2	8	10	100	118	144	118	74	1	3	64	3	2	144	1	1	64	3	2
17	2	7	3	2	8	10	84	118	164	118	64	2	3	64	3	2	164	1	1	64	3	2
18	3	5	3	2	8	10	78	90	104	90	64	1	2	48	3	2	104	1	1	48	3	2
19	3	9	14	4	6	6	414	468	546	468	360	1	4	306	5	1	546	1	1	306	5	1
20	5	9	14	4	6	6	306	368	368	368	260	1	2	246	3	1	368	1	1	246	3	1
21	2	9	7	4	6	6	286	268	416	268	228	1	5	236	4	1	416	1	1	236	4	1
22	2	9	7	4	6	7	286	276	416	276	228	1	5	246	4	1	416	1	1	246	4	1
23	3	9	7	4	6	7	260	259	312	259	206	1	4	217	3	1	312	1	1	217	3	1
24	4	9	7	4	6	7	242	222	248	222	184	1	3	188	2	1	248	1	1	188	2	1

For better comparison, the obtained results of performed tests are depicted in Diagram no.1

ISSN : 2278 – 1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 1, Issue 3, May 2012

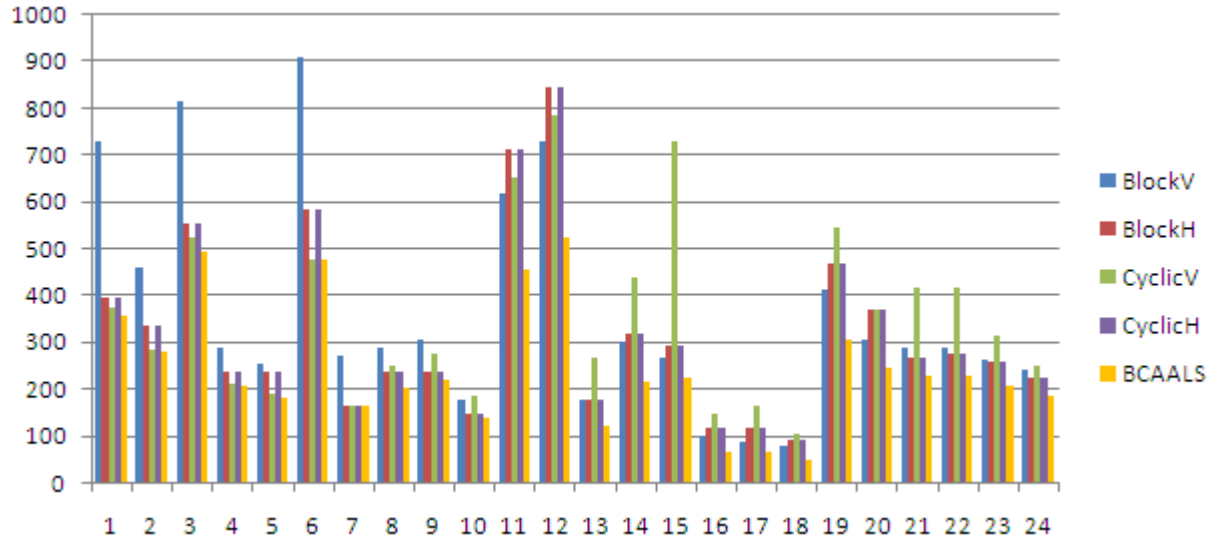

Fig. 6 The comparison of results obtained by running the proposed algorithm and former algorithms

VII. CONCLUSIONS AND FUTURE WORKS

In this paper, taking the benefits of angle shifts, an algorithm was proposed which due to its potential in improving the loading balance among processors and consequently optimizing of their usage, minimized the execution time of tiles. As he results show, the proposed algorithm (BCAALS) yielded better results in 93% of cases relative to the Block or Cyclic methods.

On the stability and reliability of the proposed algorithm it just suffices to note that as the proposed algorithm is not stochastic, its generated solutions are stable and completely (100%) reliable.

We believe that future works can focus on the generalization of this algorithm for problems with more dimensions (3-D or even more), using random algorithms for solving this problem , the parallel implementation of random algorithms such as the Genetic Algorithm and mixing the Genetic Algorithm with other intelligent searching algorithms such as Learning Automata.

REFERENCES

[1] T. Andronikos, M. Kalathas, F. M. Ciorba, P. Theodoropoulos, and G. Papakonstantinou, *An Efficient Scheduling of Uniform Dependence Loops*, Computing Systems Laboratory, Department of Electrical and Computer Engineering, National Technical University of Athens, Zographou Campus, Athens, Greece, pp. 1-10, 2004.

[2] M. Athanasaki, E. Koukis, and N. Koziris *Scheduling of Tiled Nested Loops onto a Cluster with a Fixed Number of SMP Nodes*, 12th Euromicro Conference on Parallel, Distributed and Network-Based Processing, IEEE, 2004.

[3] M. Athanasaki, A. Sotiropoulos, G. Tsoukalas, N. Koziris, and P. Tsanakas, *Hyperplane Grouping and Pipelined Schedules: How to Execute Tiled Loops Fast on Clusters of SMPs*, Journal of Supercomputing, Vol. 32, pp. 197-226, 2005.

[4] D. K. Chen And P. Ch. Yew, *An Effective Execution of Non-Uniform Do Across Loops*, IEEE, pp. 1-6, February 1995.

[5] A. Darte, L. Khachiyan, and Y. Robert, *Linear Scheduling Is Nearly Optimal*, Par. Proc. Letters, pp. 73-81, 1991.

[6] A. Darte, Y. Robert, and F. Vivien, *Scheduling and Automatic Parallelization*, Birkhäuser, 2000.

[7] I. Drositis, T. Andronikos, A. Kokorogiannis, G. Papakonstantinou, and N. Koziris, *Geometric Pattern Prediction and Scheduling of Uniform Dependence Loops*, In 5th Hellenic European Conference on Computer Mathematics and its Applications - HERCMA 2001, Athens, 2001.

[8] G. Goumas, A. Sotiropoulos, and N. Koziris, *Minimizing Completion Time for Loop Tiling with Computation and Communication Overlapping*, IEEE, pp. 1-10, 2001.

[9] A. Hajieskandar, Sh. Lotfi, *Using an Evolutionary Algorithm for Scheduling of Two-Level Nested Loops*, International Conference on Information and Electronics Engineering, pp. 100-105, May 2011.

[10] A. Hajieskandar, Sh. Lotfi, *Parallel Loop Scheduling Using an Evolutionary Algorithm*, 3rd International Conference on Advanced Computer Theory and Engineering, pp. 314-319, August 2010.

[11] L. Lamport, *The Parallel Execution of Do Loops*, Comm. of the ACM, Vol. 37, No. 2, pp. 83-93, February 1974.

[12] Sh. Lotfi and S. Parsa, *Parallel Loop Generation and Scheduling*, Journal of Supercomputing, Vol. 50, No 3, pp. 289-306, 2009.

[13] N. Manjikian and T. S. Abdelrahman, *Scheduling of Wavefront Parallelism on Scalable Shared-Memory Multiprocessors*, Department of Electrical and Computer Engineering, University of Toronto, Toronto, Canada, pp. 1-10, 1996.

[14] D. E. Maydan, J. L. Hennessy and M. S. Lam, *Efficient and Exact Data Dependence Analysis*, ACM SIGPLAN'91 Conference on Programming Language Design and Implementation, Toronto, Ontario, Canada, pp. 1-10, June 1991.

[15] D. I. Moldovan and J. Fortes, *Partitioning and Mapping Algorithms into Fixed Size Systolic Arrays*, IEEE Transactions on Computers, Vol. C-35, No. 1, pp. 1-11, 1986.

[16] T. W. O'Neil, *Techniques for Optimizing Loop Scheduling*, Ph. D. Thesis, The Graduate School of the University of Notre Dame, Indiana, 2002.

[17] S. Parsa, and Sh. Lotfi, *A New Approach to Parallelization of Serial Nested Loops Using Genetic Algorithms*, Journal of Supercomputing, Vol. 36, No. 1, pp. 83–94, 2006.

[18] S. Parsa, and Sh. Lotfi, *A New Genetic Algorithm for Loop Tiling*, Journal of Supercomputing, Vol. 37, No. 3, pp. 249–269, 2006.
[19] S. Parsa, and Sh. Lotfi, *Code Generation and Scheduling for Parallelization of Multi-Dimensional Perfectly Nested Loops*, 12th International CSI Computer Conference (CSICC'07) Shahid Beheshti University, Tehran, Iran, pp. 20-22, February 2007.
[20] S. Parsa and Sh. Lotfi, *Wave-Fronts Parallelization and Scheduling*, IEEE, 4th International Conference on Innovations in Information Technology (Innovations'07), Dubai, UAE, pp. 382-386, November 18-20, 2007.
[21] D. L. Pean, H. T. Chua, and CH. Chen, *A Release Combined Scheduling Scheme for Non-Uniform Dependence Loops*, Journal of Information Science and Engineering, Vol. 18, pp. 223-255, 2002.
[22] F. Rastello, and Y. Robert, *Automatic Partitioning of Parallel Loops with Parallelepiped-Shaped Tiles*, IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 5, pp. 460-470, May 2002.
[23] W. Shang and J. A. B. Fortes, *Time Optimal Linear Schedules for Algorithms with Uniform Dependencies*, IEEE Transactions on Computers, Vol 40, No. 6, pp. 723-742, 1991.
[24] J. Xue, *On Tiling as a Loop Transformation*, Parallel Processing Letters, Vol. 7, No. 4, pp. 409-424, 1997.
[25] Ch. T. Yang, and K. Cheng, *An Enhanced Parallel Loop Self-Scheduling Scheme for Cluster Environments*, Journal of Supercomputing, Vol. 34, pp. 315-335, 2005.

Biography

AliReza Hajieskandar received the B.Sc. in Software Engineering from Islamic Azad University shabestar Branch, Iran and the M.Sc. degree in Software Engineering from Islamic Azad University Qazvin Branch, Iran. He is preceptor of Software Engineering at the Islamic Azad University Bonab Branch. His research interests include compilers, super-compilers, parallel processing, evolutionary computing and algorithms.

Shahriar Lotfi received the B.Sc. in Software Engineering from the University of Isfahan, Iran, the M.Sc. degree in Software Engineering from the University of Isfahan, Iran, and the Ph.D. degree in Software Engineering from Iran University of Science and Technology in Iran. He is Assistant Professor of Computer Science at the University of Tabriz. His research interests include compilers, super-compilers, parallel processing, evolutionary computing and algorithms.

Simin Ghahramanian received the B.Sc. in Software Engineering from Islamic Azad University shabestar Branch, Iran, she is studying in M.Sc. degree in computer system architecture.she is preceptor of Software Engineering at the Sama Technical and Vocational Training College, Islamic Azad University, Bonab Branch. His research interests include parallel processing, Network Computing and NOC Testing